**“Baithak”**

**18CSC302J – Computer Networks**

**Mini Project Report**

*Submitted by*

**Harshit Kumar [RA2111027010003]**

**Safal Mehrotra [RA2111027010006]**

**Gaurang Ashava [RA2111027010007]**

**Syed Adnan Hussainy [RA2111027010008]**

**Ansh Aggarwal [RA2111027010042]**

****

**SCHOOL OF COMPUTING**

**COLLEGE OF ENGINEERING AND TECHNOLOGY**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**(Under Section 3 of UGC Act, 1956)**

S.R.M. NAGAR, KATTANKULATHUR – 603 203

CHENGALPATTU DISTRICT

**November 2023**

****

**COLLEGE OF ENGINEERING AND TECHNOLOGY**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**(Under Section 3 of UGC Act, 1956)**

S.R.M. NAGAR, KATTANKULATHUR – 603 203

**BONAFIDE CERTIFICATE**

Certified that Mini project report titled Baithak is the bonafide work of Reg.No RA2111027010003, RA2111027010004, RA2111027010007, RA2111027010008 & RA2111027010042 Name Harshit Kumar, Safal Mehrotra, Gaurang Ashava, Syed Adnan Hussainy & Ansh Aggarwal who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**SIGNATURE SIGNATURE**

**(GUIDE) (HEAD OF THE DEPARTMENT)**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S No.** | **Title** | **Page No.** |
| 1 | Abstract | 3 |
| 2 | Problem Statement | 4 |
| 3 | Methodology / Procedure/ Algorithm | 5 |
| 4 | Front-end code (HTML, CSS, Javascript) | 6 |
| 5 | Modules of the proposed work | 10 |
| 6 | Results/Screenshots | 12 |
| 7 | Conclusion | 13 |
| 8 | References | 14 |

1. **Abstract**

This project revolves around the development of a real-time chat application using Socket.IO, tailored for concurrent interactions among numerous users. Rooted in computer networks principles, the Node.js-based server orchestrates socket connections, allowing users to engage in global chat, create private rooms, and seamlessly communicate in real-time. Socket.IO's bidirectional communication ensures low-latency exchanges, enabling dynamic interactions within a multi-user environment.

The application's core features include user connection management, global chat functionality, and the creation and joining of private rooms. Users can communicate globally or within specific rooms, enhancing privacy and enabling targeted discussions. The server effectively handles user disconnections, ensuring a robust and user-friendly experience. This project lays the groundwork for scalable, real-time communication applications, offering a versatile platform for collaborative discussions among diverse user groups.

Moreover, the project leverages a client-server architecture that promotes flexibility and extensibility. The server-side code efficiently manages the complexities of room creation, user interactions, and disconnections. On the client side, the application's user interface is crafted using HTML, CSS, and JavaScript, with Socket.IO seamlessly integrating real-time updates. This design not only facilitates a fluid user experience but also sets the stage for potential future enhancements, such as multimedia sharing, user authentication, or additional interactive features, further solidifying the application's potential as a comprehensive and adaptable real-time communication platform.

Additionally, this project addresses the contemporary demand for secure and scalable real-time communication platforms. By employing Socket.IO and Node.js, the application aims to overcome the limitations of traditional chat systems, providing a dynamic environment where users can not only exchange messages globally but also create personalized rooms for more focused discussions. The emphasis on user connection management ensures a reliable and uninterrupted experience, while the client-server architecture and the incorporation of web technologies pave the way for future feature expansions and optimizations. In essence, this project seeks to deliver a cutting-edge solution that redefines the standards for multi-user chat applications in the era of instantaneous digital communication.

1. **Problem Statement**

The problem at hand revolves around the need for an advanced real-time multi-user chat application that addresses the limitations of existing solutions in terms of scalability, responsiveness, and user engagement. Present chat platforms often struggle to efficiently manage concurrent interactions among a multitude of users, hindering seamless communication and collaboration.

This project aims to develop a robust solution using Socket.IO and Node.js, leveraging computer network principles to establish a dynamic and secure environment. The specific challenges include implementing global chat functionality, enabling users to create and join private rooms, and ensuring effective handling of user connections and disconnections. This application seeks to redefine the standards of real-time communication platforms by delivering a comprehensive and adaptable solution that caters to the evolving needs of diverse user groups in the digital age. The successful completion of this project would signify a significant advancement in real-time communication applications, offering a versatile platform for collaborative discussions, targeted messaging, and future feature expansions.

**3. Methodology**

The methodology for this real-time multi-user chat application project involves a systematic approach to design, development, and implementation. Initially, a comprehensive analysis of the project requirements and specifications is conducted to define the application's features and functionalities. This includes delineating the user interactions, messaging protocols, and room management.

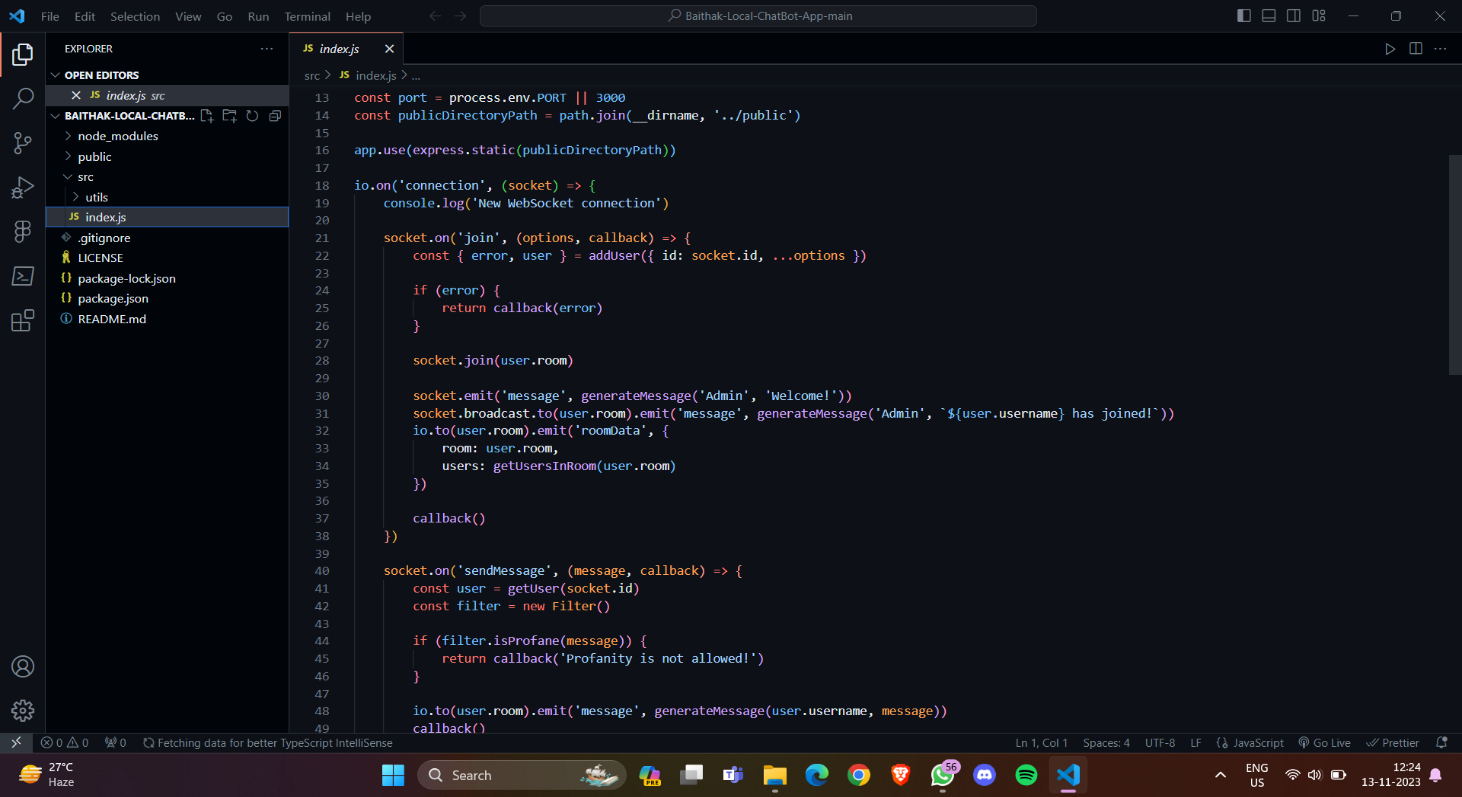
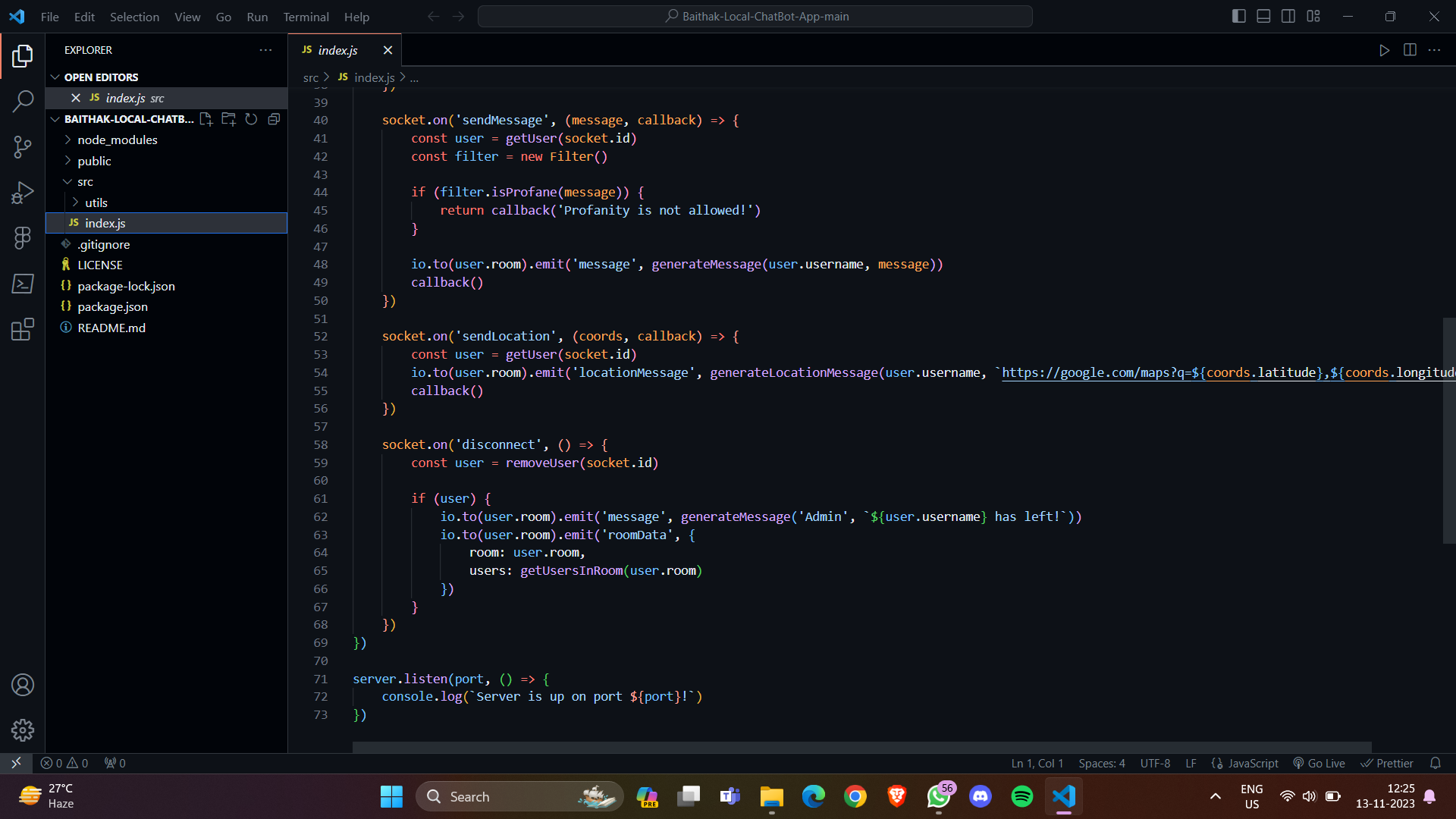
Following the analysis, the project proceeds with the design phase, where the architecture and components of the server-client system are outlined. Special attention is given to the integration of Socket.IO for enabling real-time bidirectional communication. The server is structured to handle user connections, chat messaging, and room management efficiently. Simultaneously, the client-side interface, constructed using HTML, CSS, and JavaScript, is designed to provide an intuitive user experience.

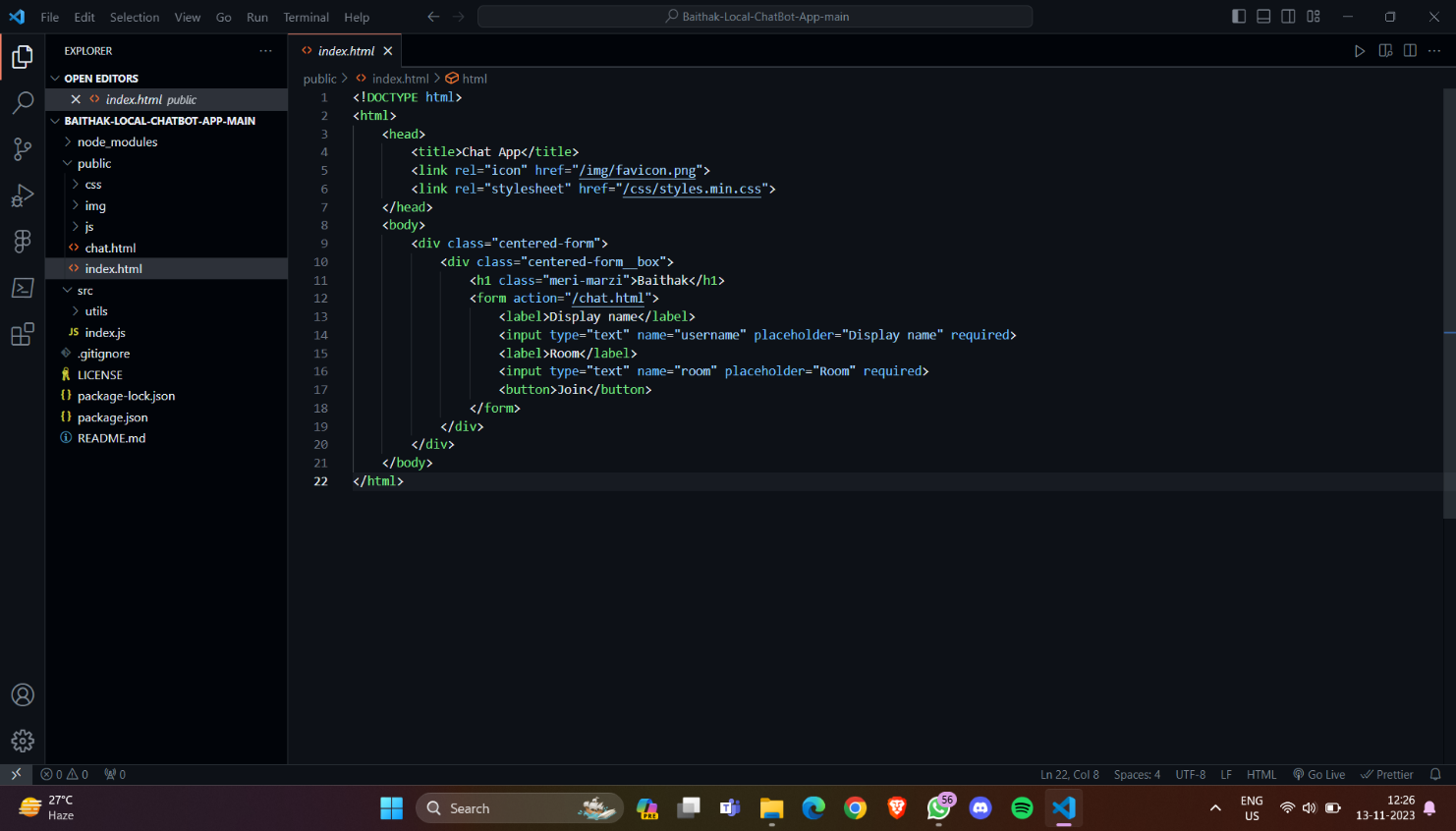
The development phase involves the implementation of the server and client code. Node.js is utilized for the server-side logic, and Socket.IO ensures seamless communication between the server and clients. This phase also includes testing to validate the application's functionality and identify and rectify any potential issues.

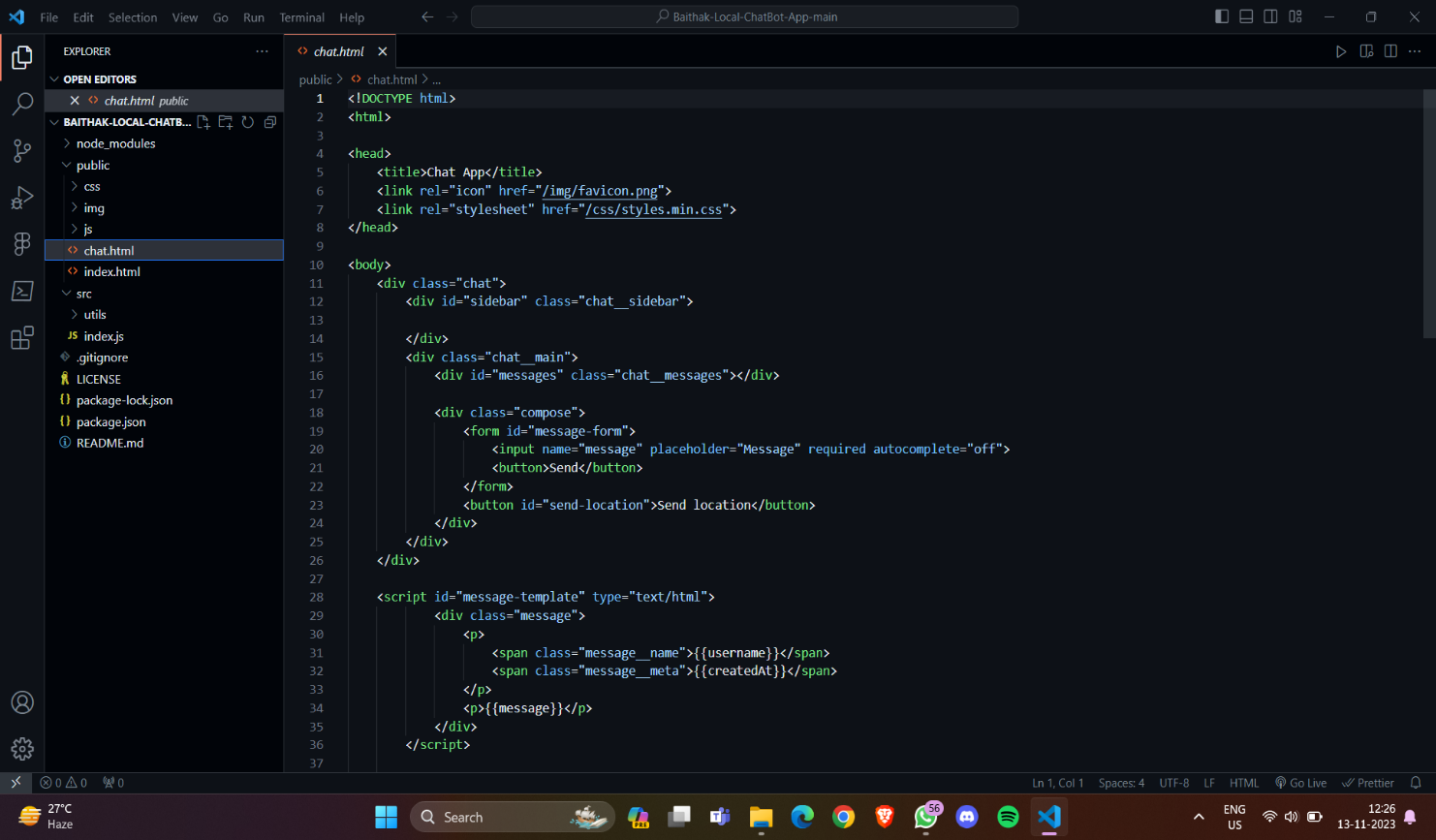
The iterative nature of the development process allows for continuous refinement based on testing feedback. Finally, the deployment phase ensures that the application is ready for use on a designated server. The methodology emphasizes an agile development approach, ensuring adaptability to changing requirements and providing a solid foundation for future feature enhancements and optimizations.

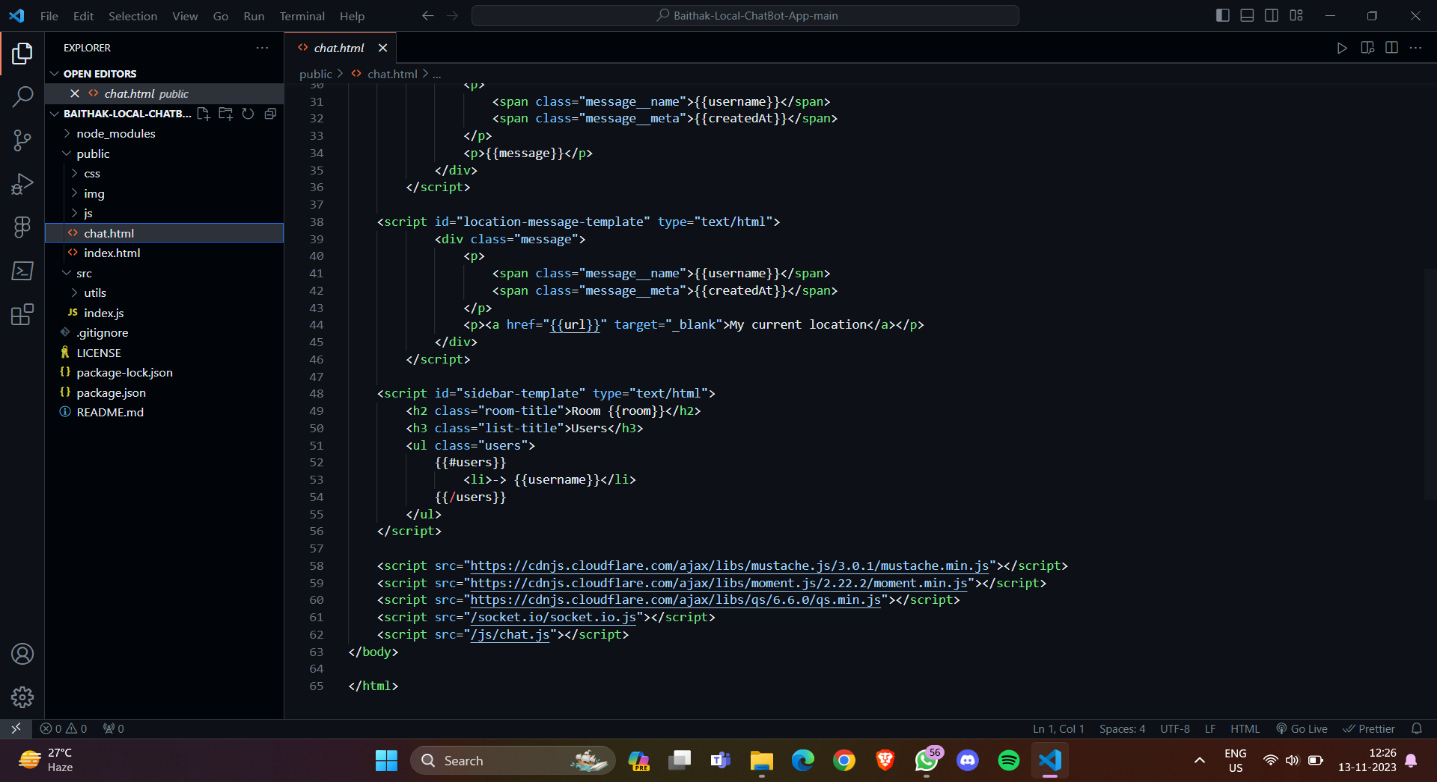
**4. Front-end code (HTML, CSS, JavaScript)**

* Socket Code:



* HTML Code:





**5. Modules of the proposed work**

1. User Management Module:

- Handles user connections and disconnections.

- Manages user metadata (e.g., username, socket ID).

- Logs user activities and status changes.

2. Global Chat Module:

- Facilitates global messaging among all connected users.

- Handles the broadcasting of messages to all connected clients.

3. Room Management Module:

- Manages the creation and joining of private rooms.

- Handles room-specific messaging.

- Logs room activities and status changes.

4. Message Handling Module:

- Receives and processes incoming messages from users.

- Determines the type of message (global or room-specific).

- Routes messages to the appropriate module for broadcasting.

5. Socket.IO Integration Module:

- Integrates Socket.IO for real-time bidirectional communication.

- Manages socket events, such as connection and disconnection.

6. Server-Side Processing Module:

- Orchestrates the overall server-side logic.

- Coordinates interactions between different modules.

- Handles initialization and shutdown procedures.

7. Client-Side Interface Module:

- Develops the user interface using HTML, CSS, and JavaScript.

- Integrates with Socket.IO for real-time updates.

- Handles user input and displays messages.

8. Error Handling Module:

- Identifies and manages potential errors or exceptions.

- Provides appropriate error messages or fallback mechanisms.

9. Testing and Debugging Module:

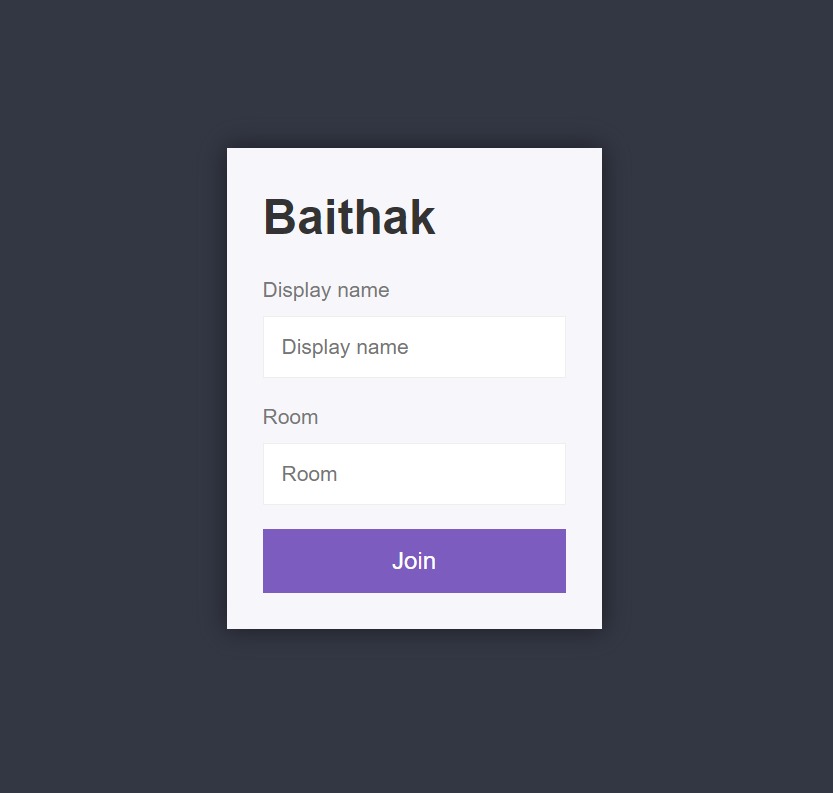
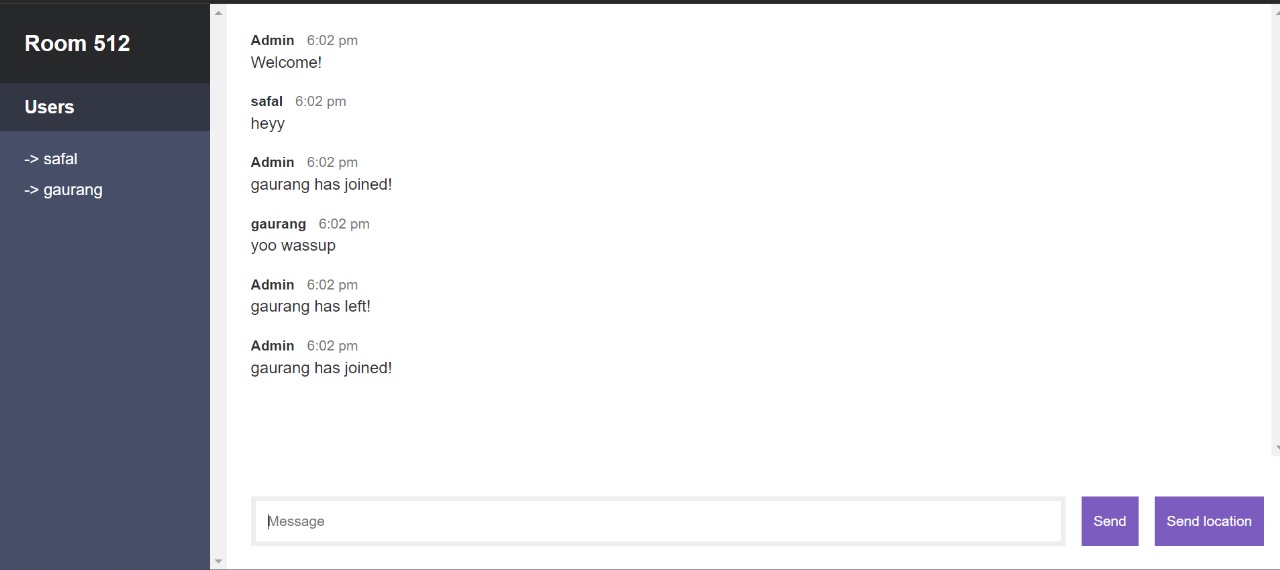
- Implements testing procedures to validate functionality.

- Facilitates debugging and troubleshooting during development.

10. Deployment and Hosting Module:

- Manages the deployment of the application on a designated server.

- Handles server hosting and configuration.

**6. Result / Screenshots**

**7. Conclusions**

In conclusion, the real-time multi-user chat application project represents a significant advancement in the realm of digital communication, addressing key challenges associated with scalability, responsiveness, and user engagement. By harnessing the capabilities of Socket.IO and Node.js, the project successfully delivers a dynamic platform where users can seamlessly connect, communicate globally, and create private rooms for more focused discussions. The modular architecture, encompassing distinct modules for user management, messaging, and room control, not only enhances the maintainability and scalability of the application but also lays the groundwork for future expansions and feature integrations.

The project's emphasis on efficient user connection management ensures a reliable and uninterrupted experience, while the real-time broadcasting of messages using Socket.IO facilitates instantaneous communication, contributing to a vibrant and interactive user environment. The incorporation of a client-side interface using HTML, CSS, and JavaScript enhances the user experience, providing an intuitive and visually appealing platform for interaction.

Furthermore, the successful implementation of room-specific messaging enhances user privacy and facilitates targeted discussions. The iterative and agile development methodology adopted throughout the project allowed for continuous refinement, ensuring that the final product meets the evolving needs of users in the digital age.

Looking forward, the real-time multi-user chat application serves as a versatile foundation for future enhancements. Possible avenues for expansion include the incorporation of multimedia sharing, user authentication, and additional interactive features. Overall, the project not only fulfills the immediate need for a robust real-time communication platform but also positions itself as a flexible and adaptive solution capable of accommodating emerging requirements in the ever-evolving landscape of digital collaboration.

**8. References**

**1.** [**https://expressjs.com/**](https://expressjs.com/)

**2.** [**https://www.w3schools.com/html/default.asp**](https://www.w3schools.com/html/default.asp)

**3.** [**https://nodejs.org/en/docs/**](https://nodejs.org/en/docs/)

**4.** [**https://socket.io/docs/v4/**](https://socket.io/docs/v4/)

**5.** [**https://www.tutorialspoint.com/websockets/index.htm**](https://www.tutorialspoint.com/websockets/index.htm)

**6.** [**https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide**](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide)

**7.** [**https://github.com/expressjs/express**](https://github.com/expressjs/express)

**8.** [**https://github.com/sorrycc/awesome-javascript**](https://github.com/sorrycc/awesome-javascript)